

FIGURE 1

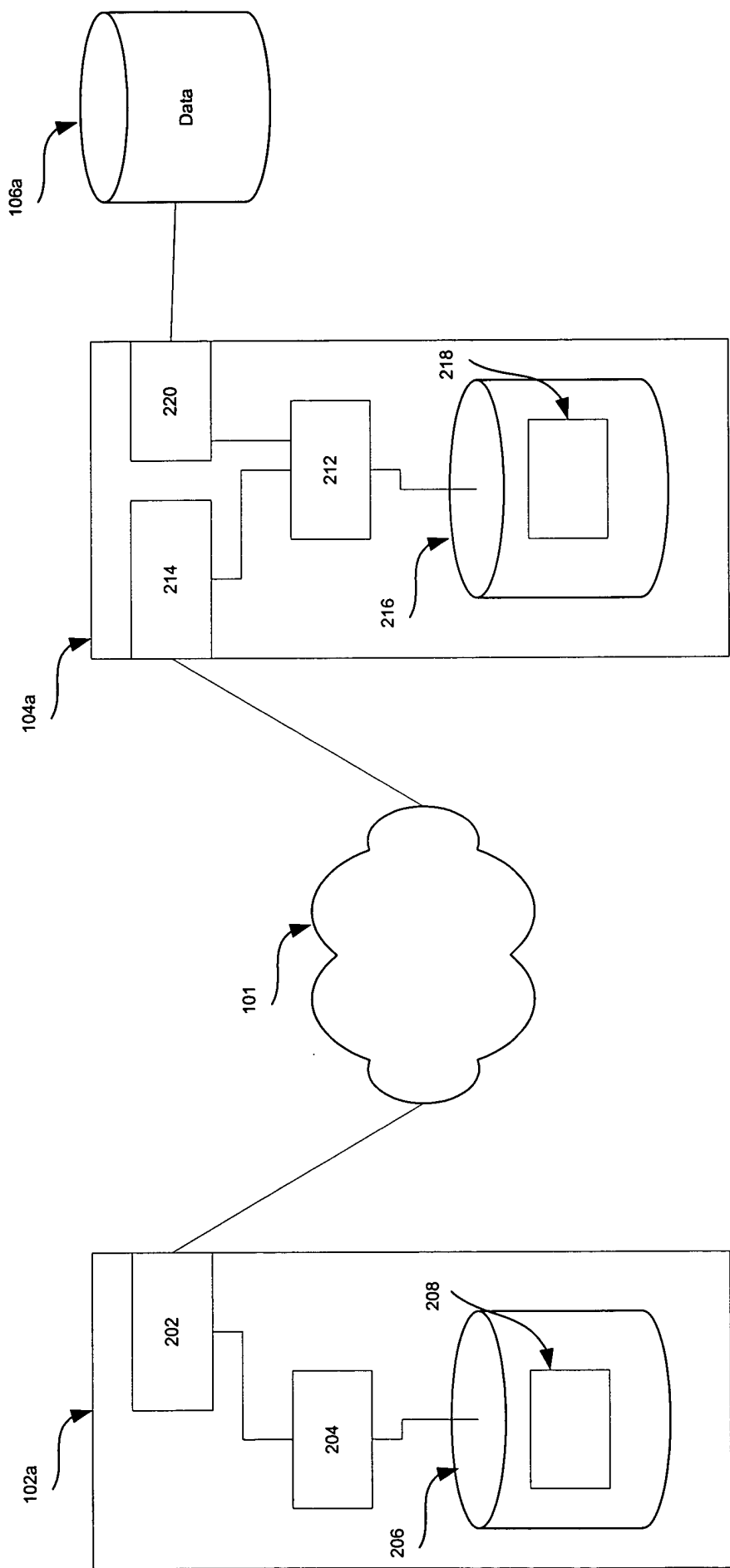


FIGURE 2

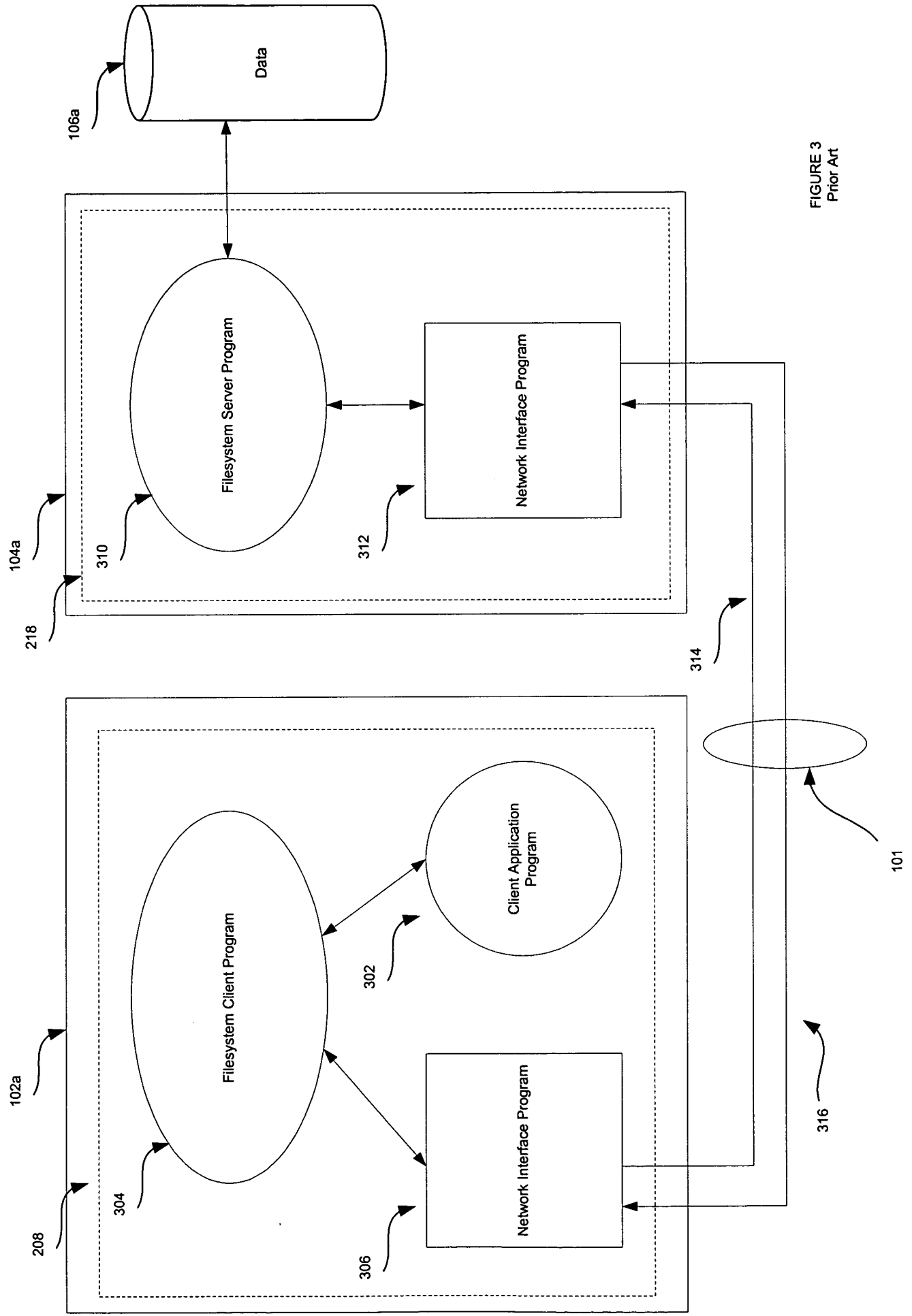


FIGURE 3
Prior Art

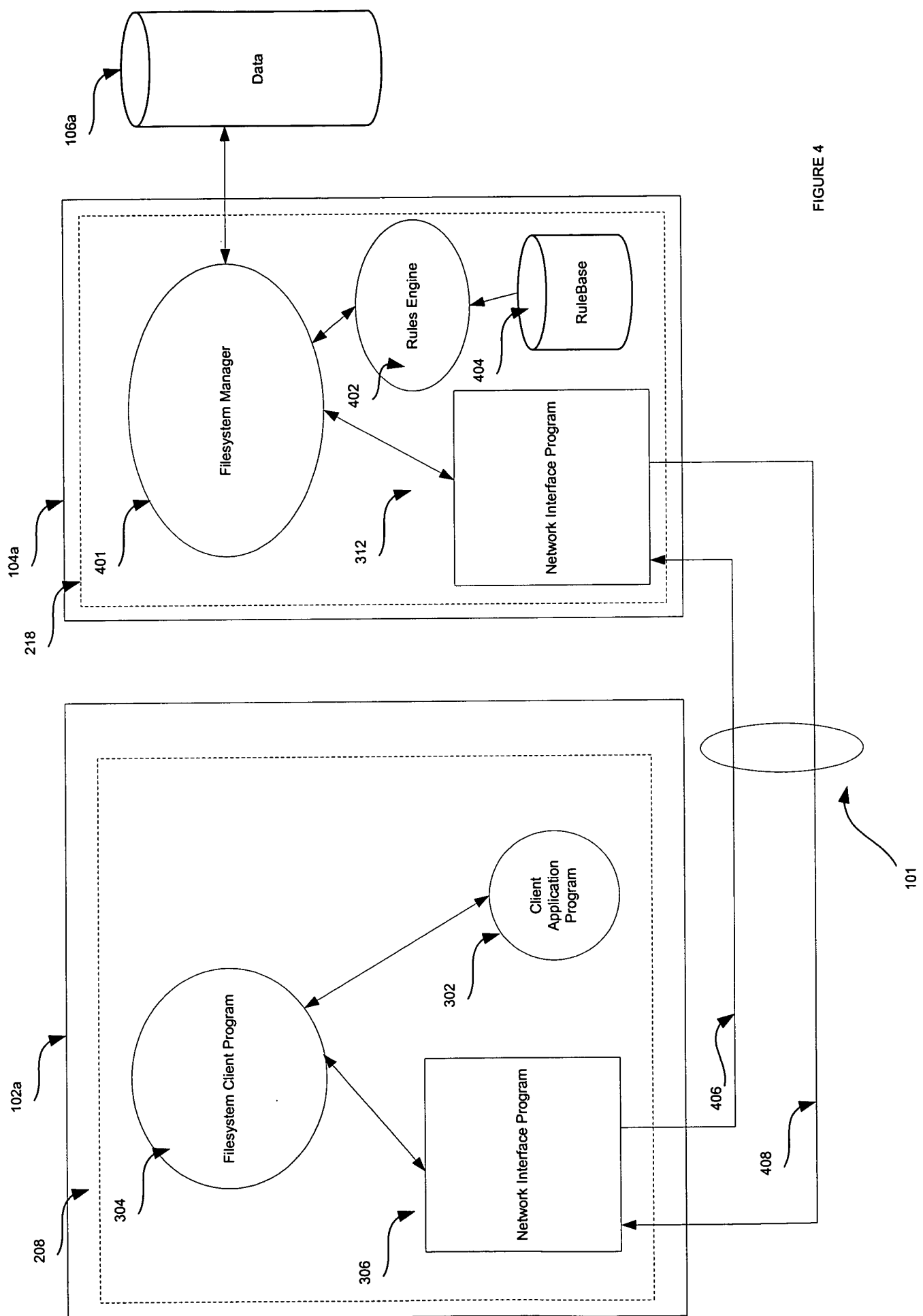


FIGURE 4

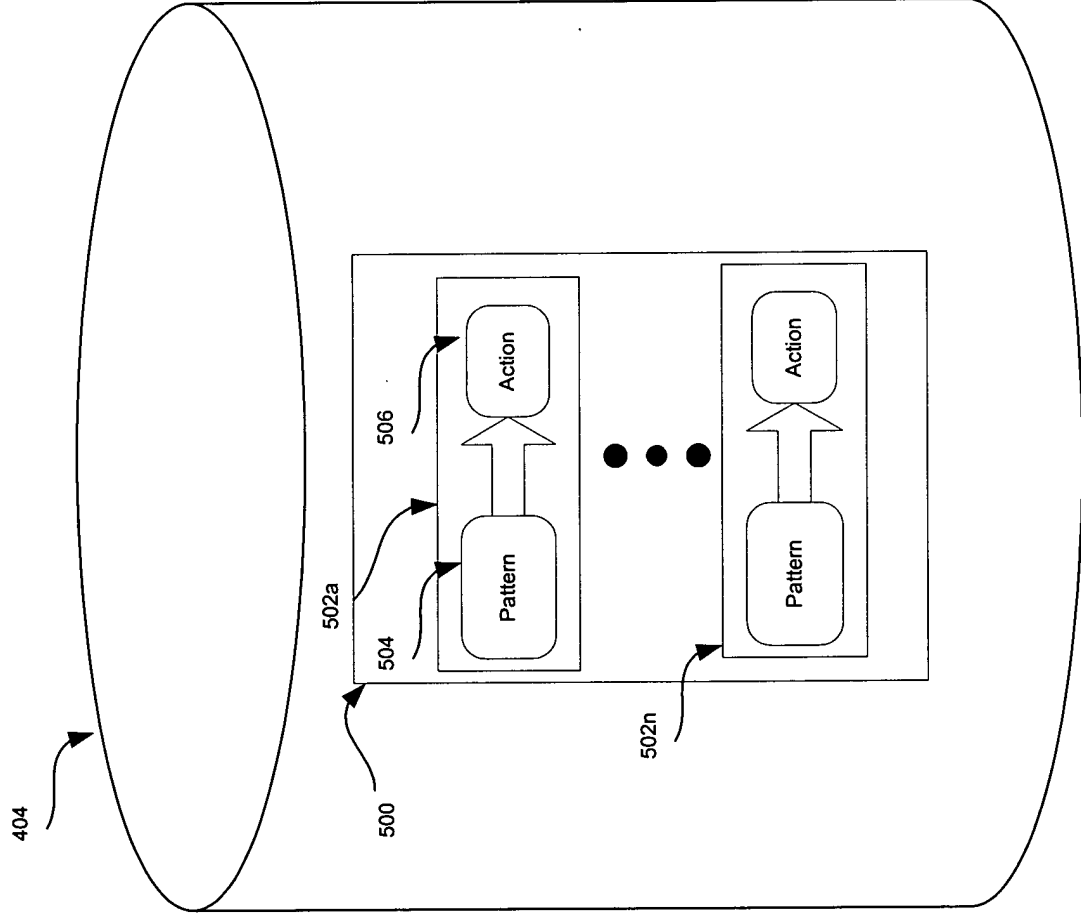


FIGURE 5

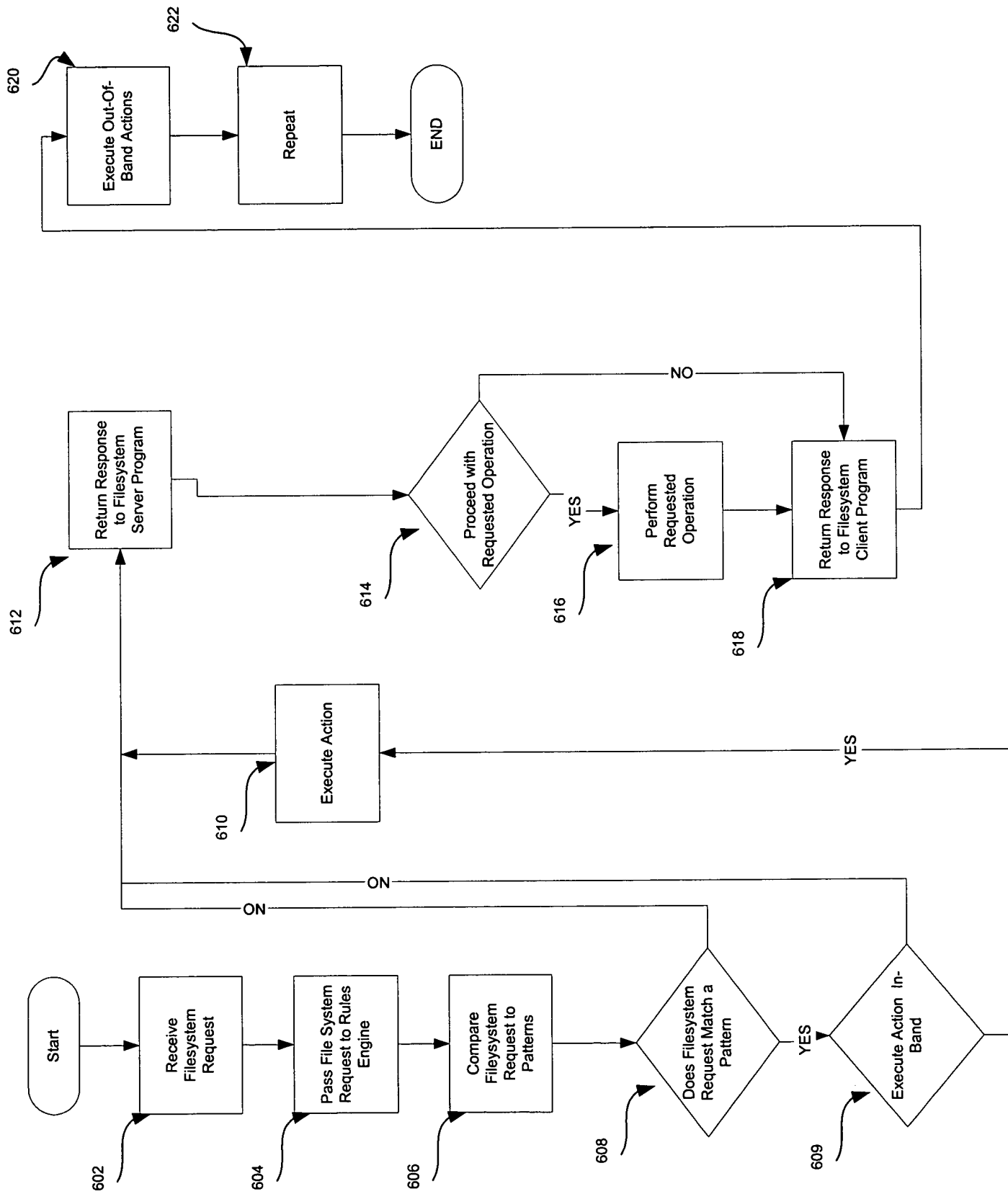


FIGURE 6

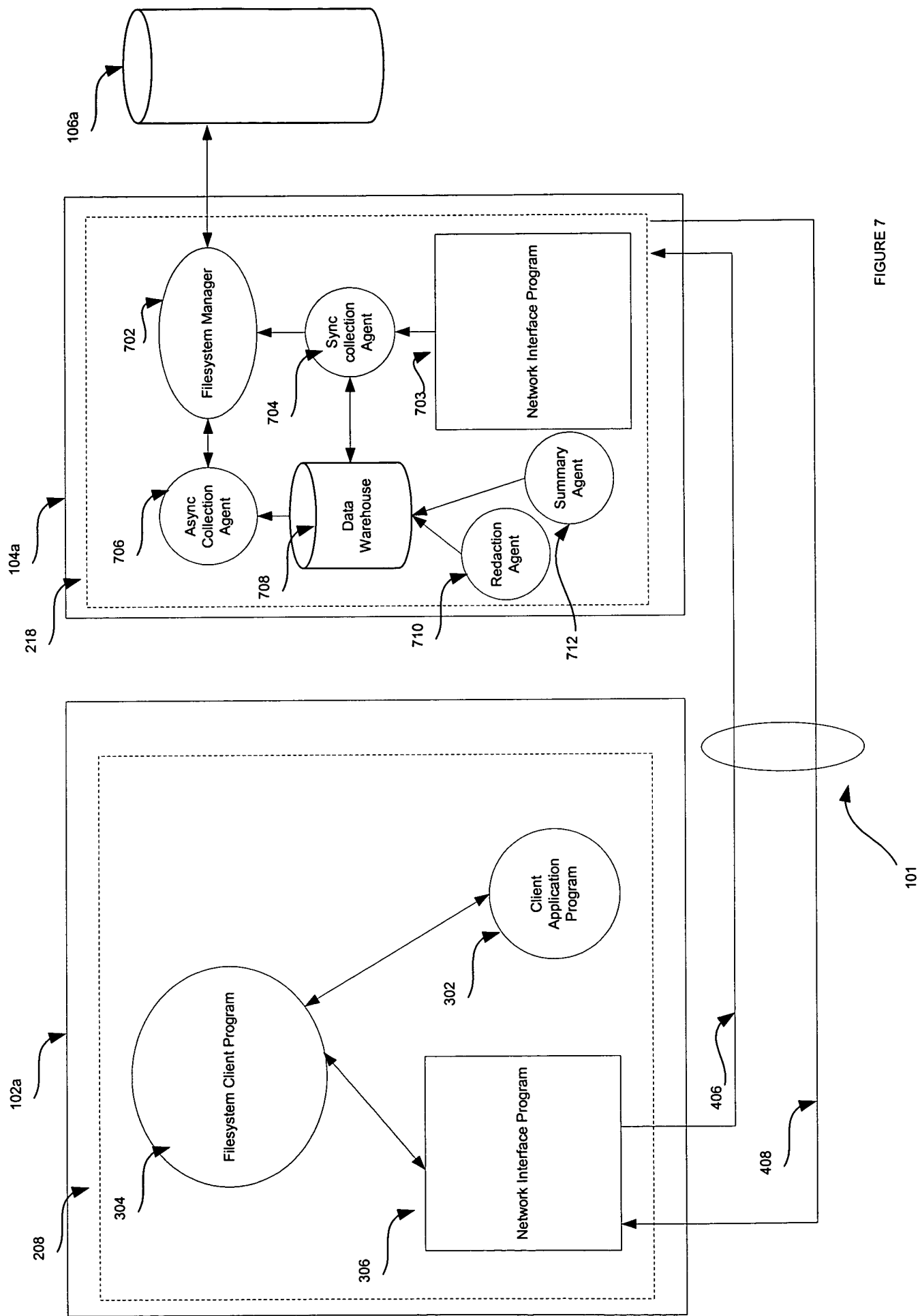


FIGURE 7

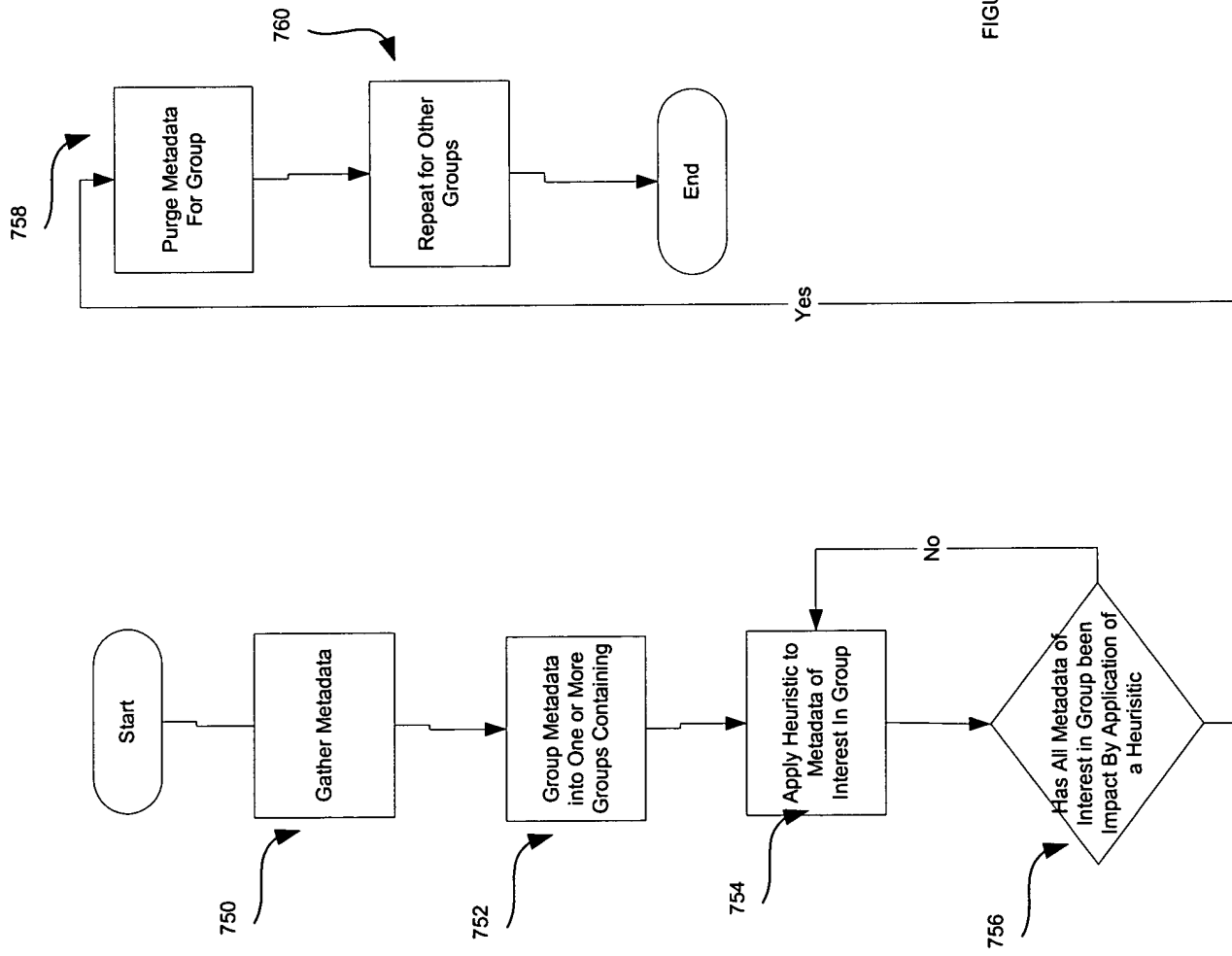


FIGURE 8

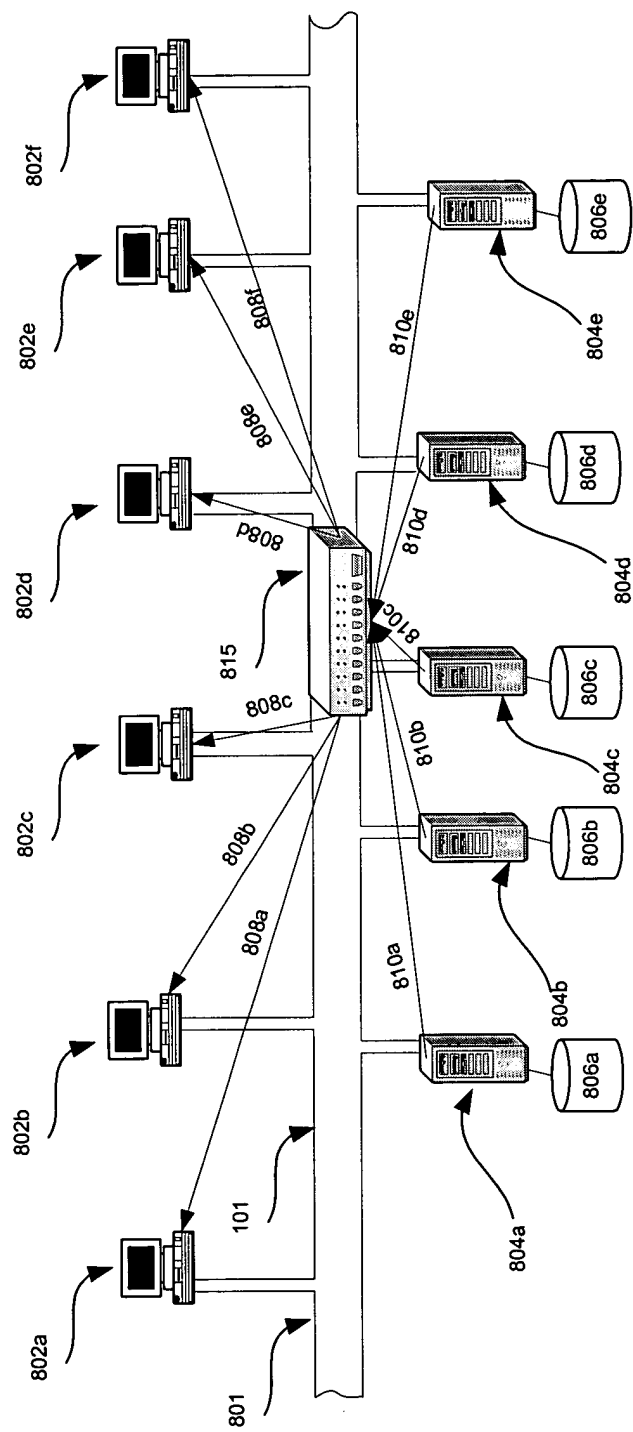


FIGURE 9

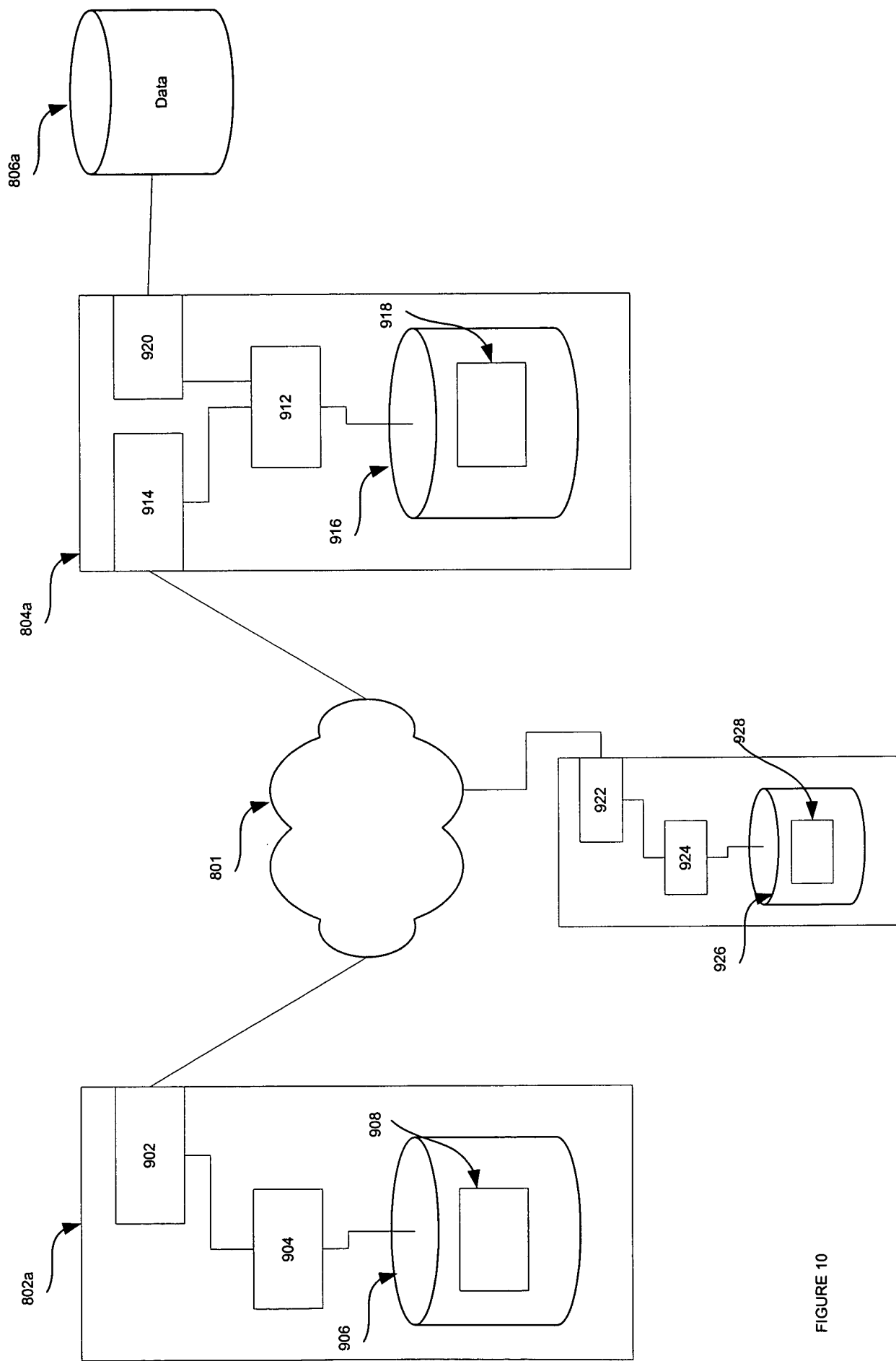


FIGURE 10

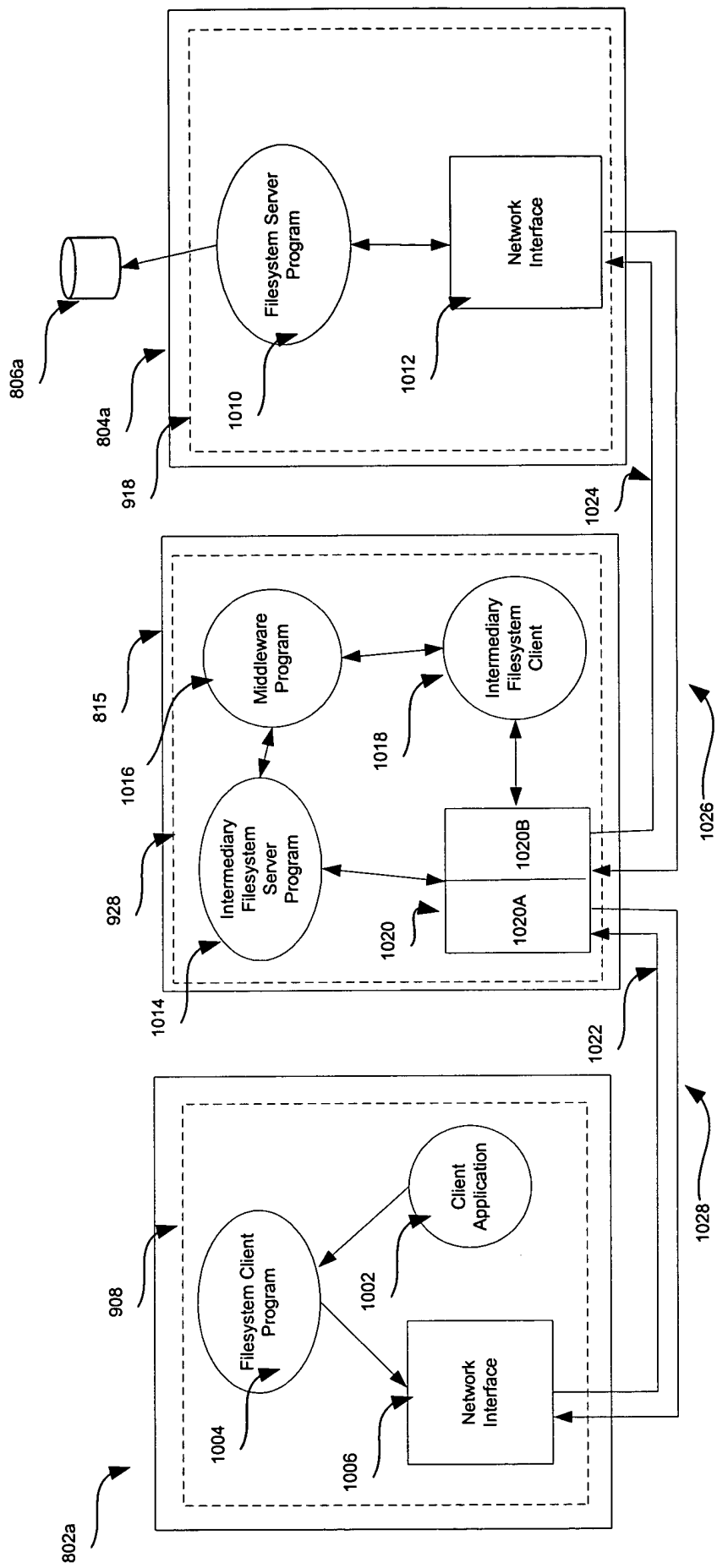


FIGURE 11

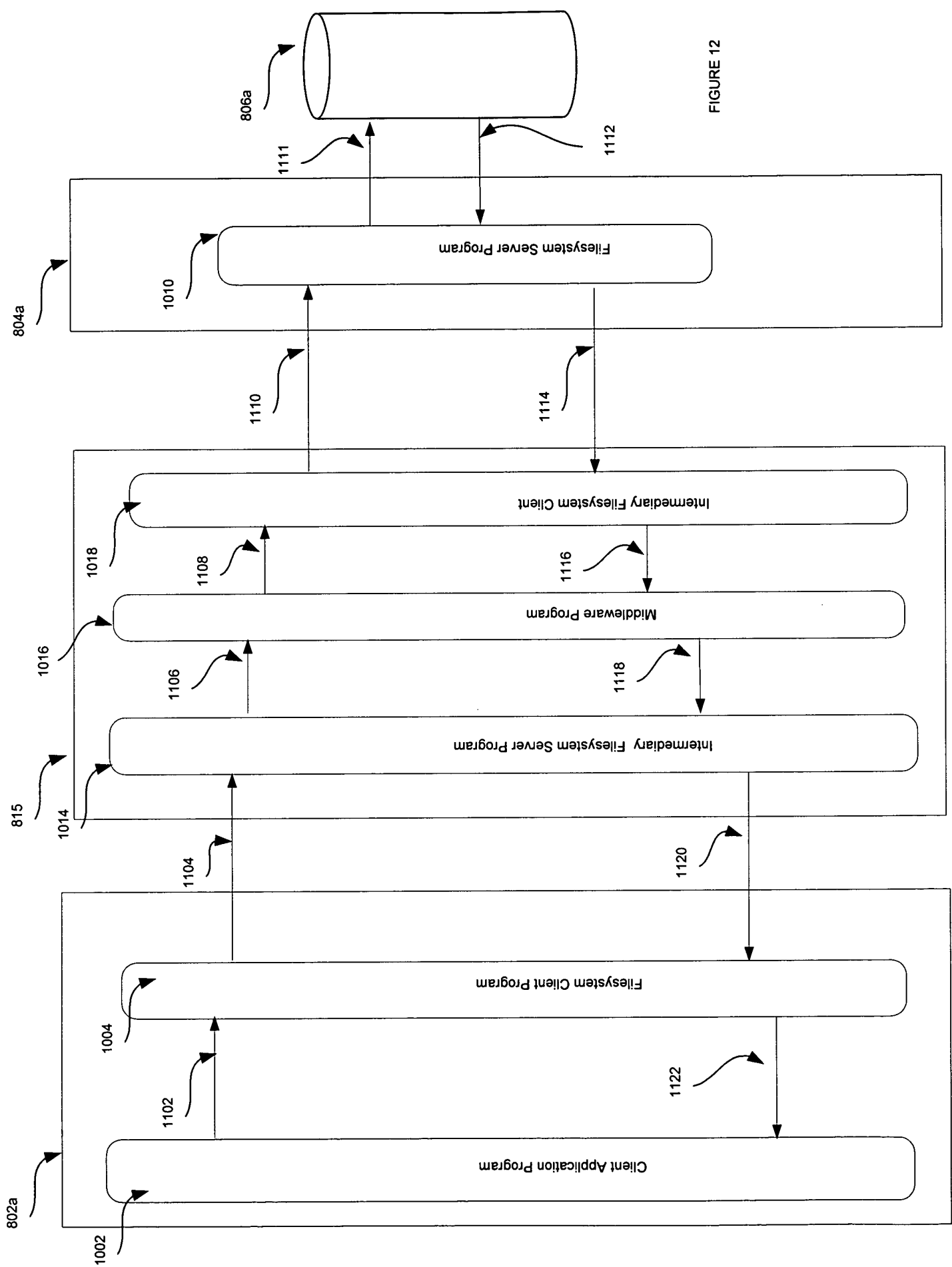


FIGURE 12

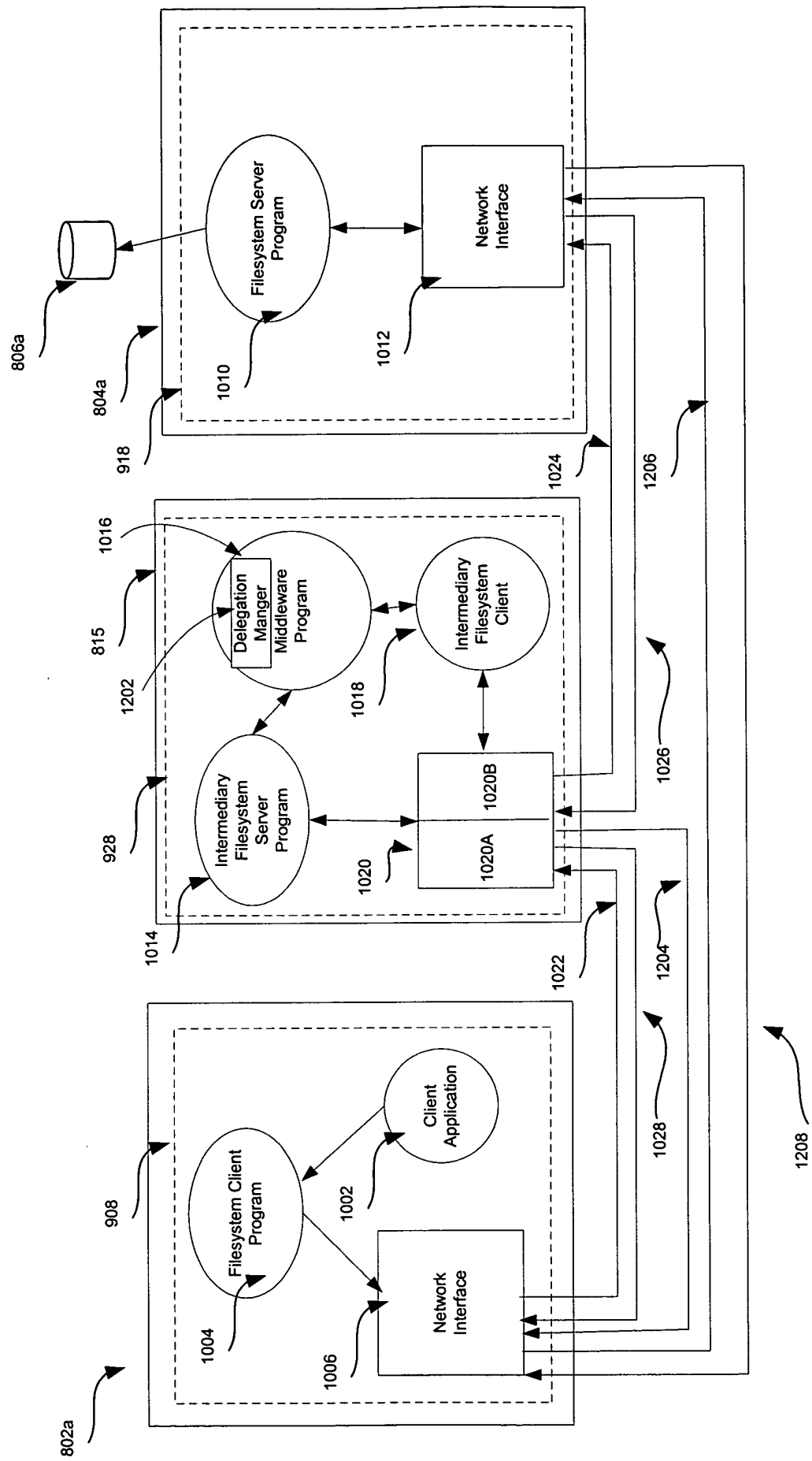


FIGURE 13

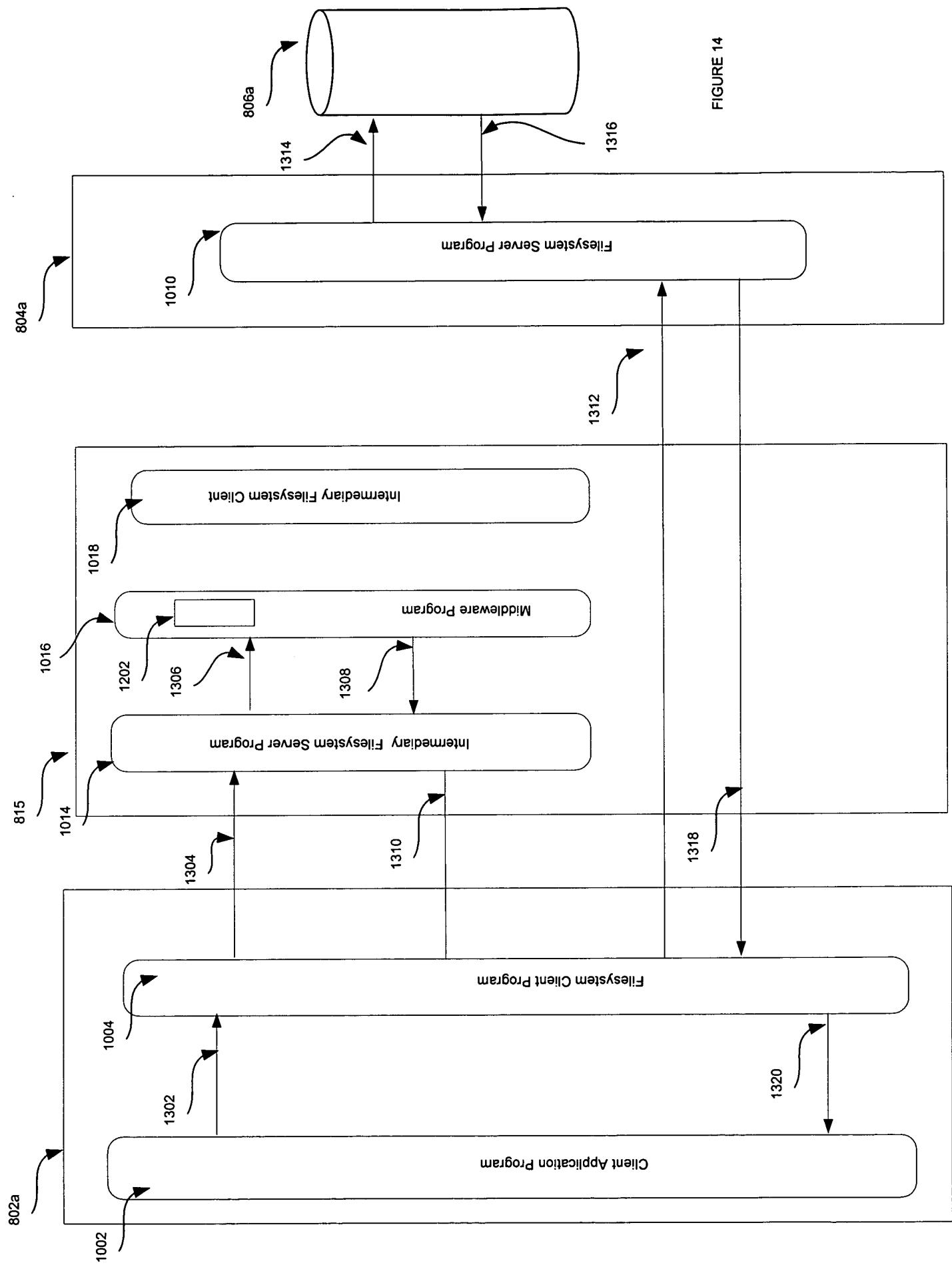


FIGURE 14

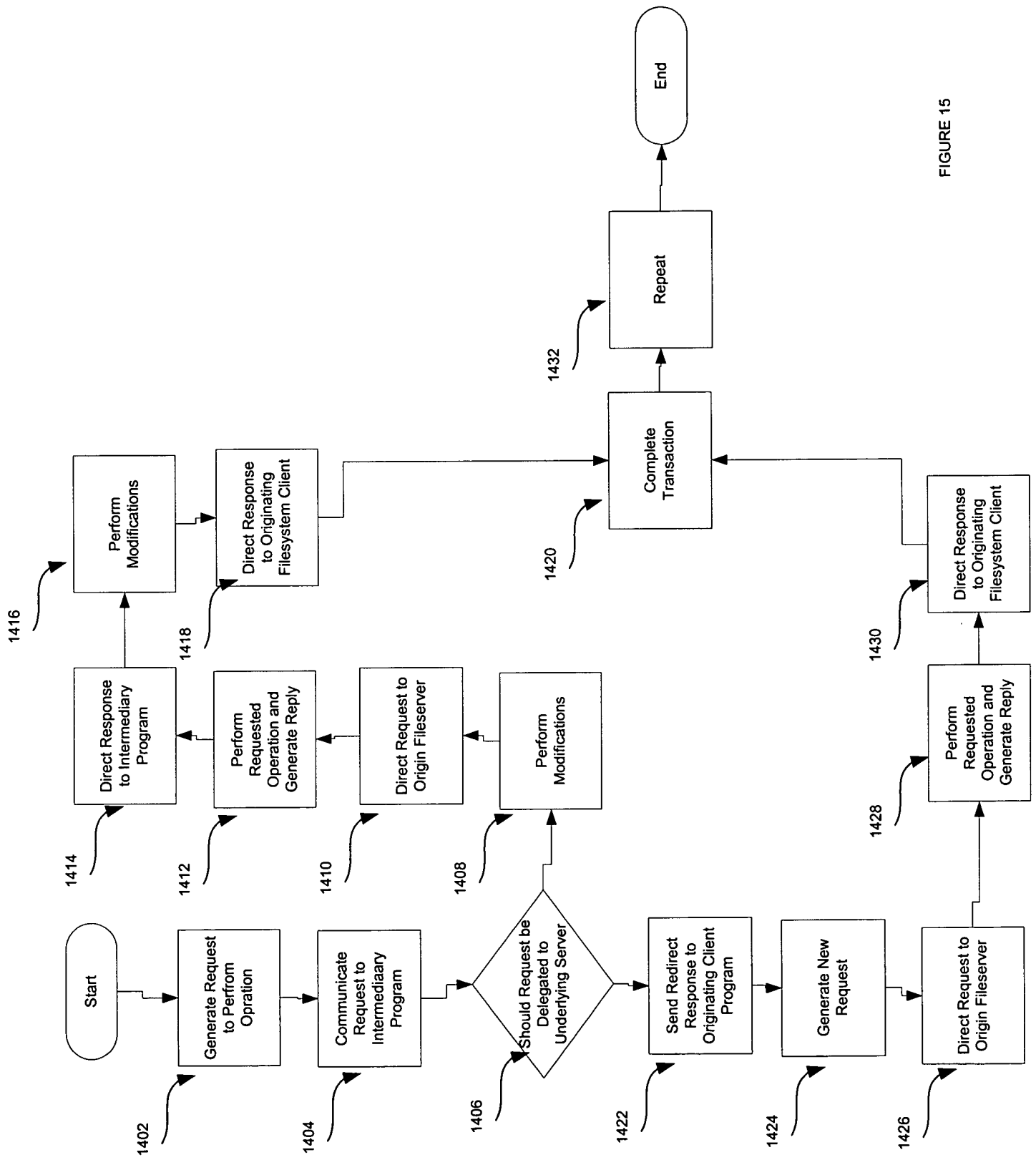


FIGURE 15

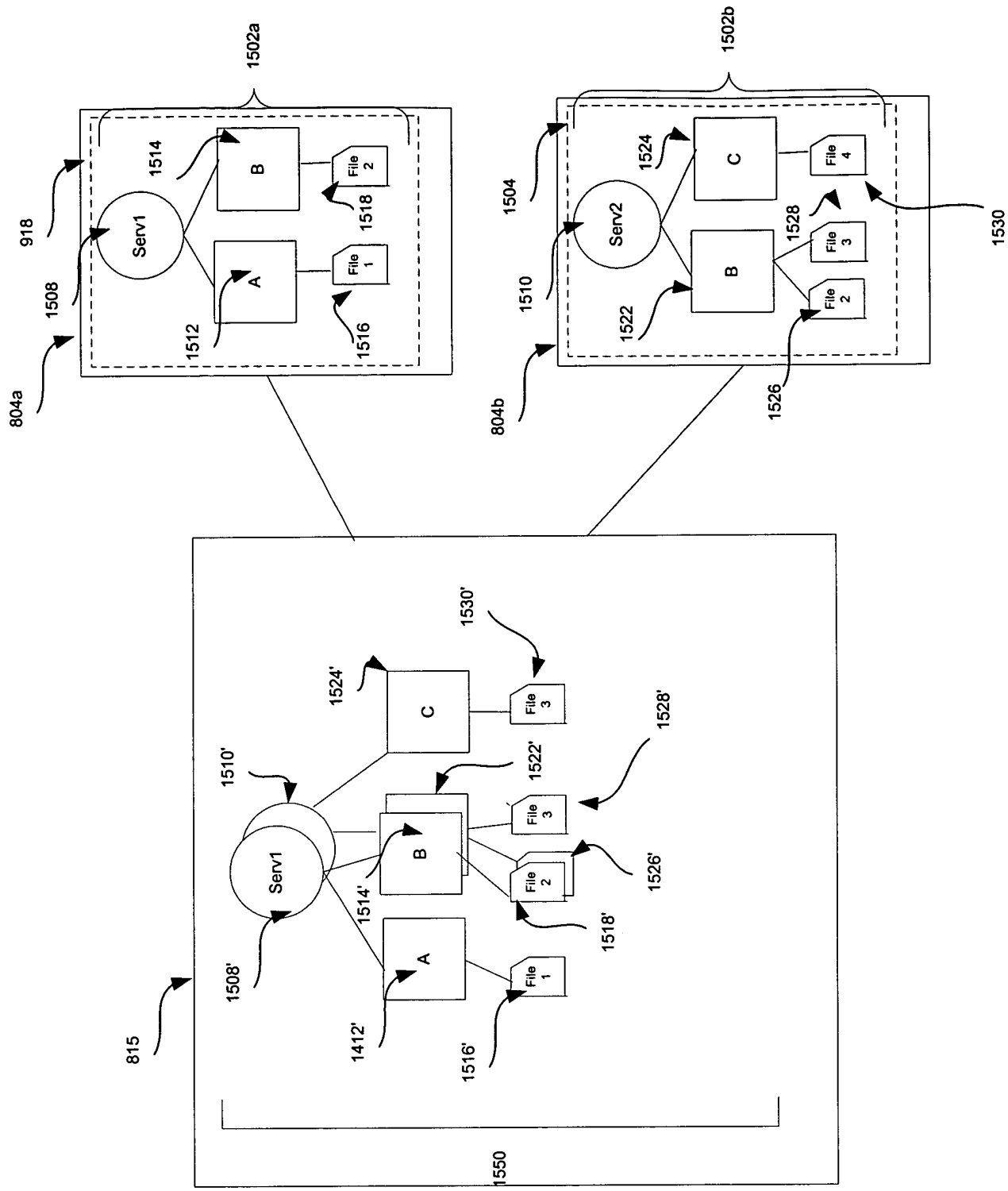


FIGURE 16

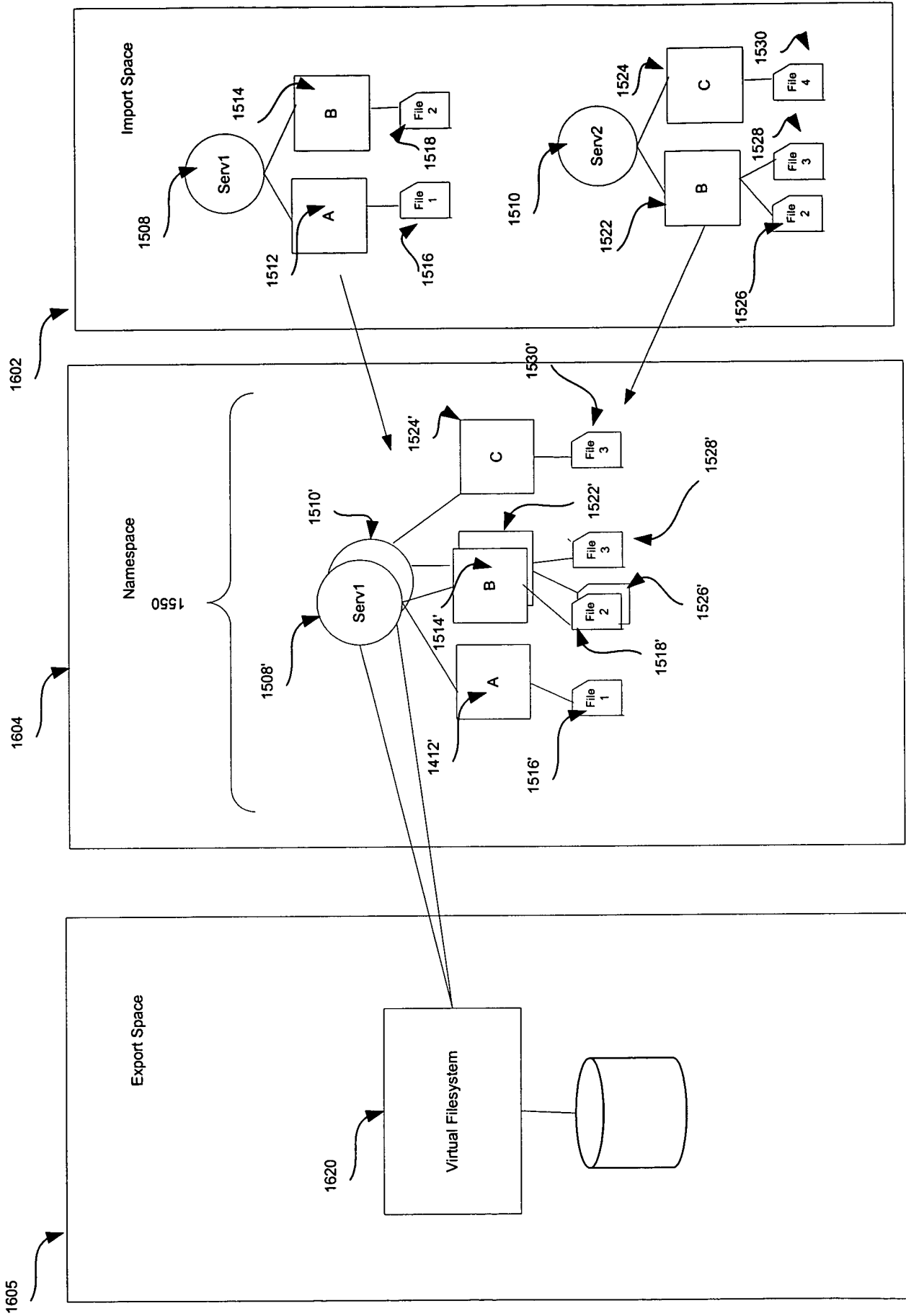


Figure 17

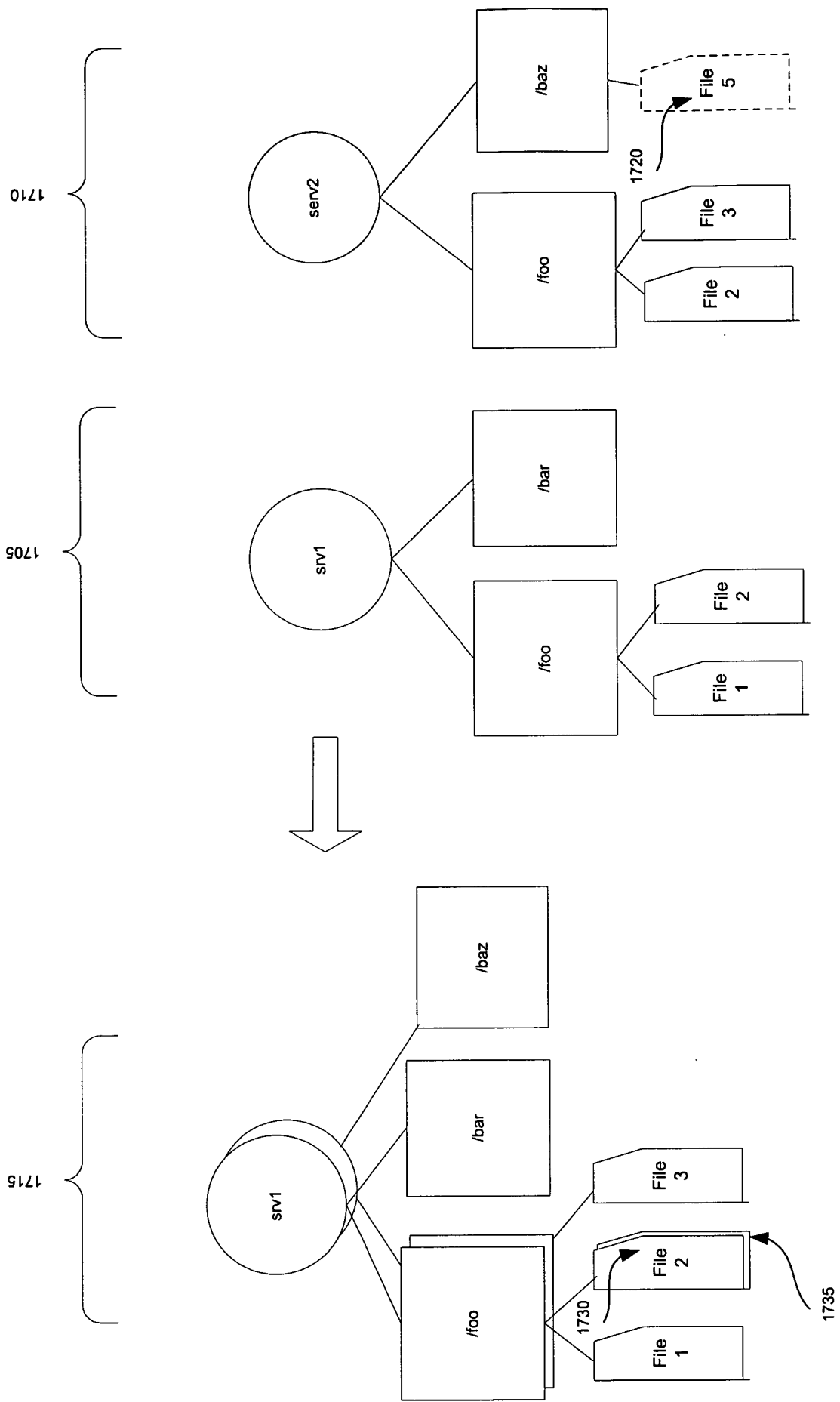


FIGURE 18

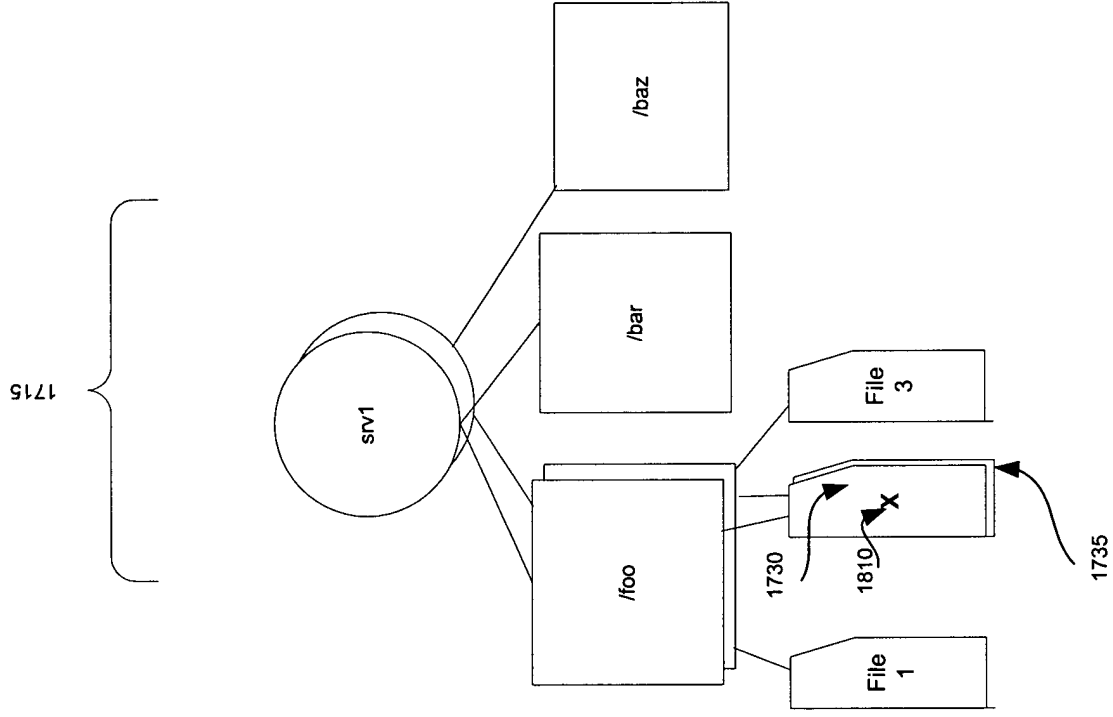


Figure 19

```

{{{
/* This module implements "write-through" semantics:
First, the operation is attempted in the topbase,
If the file/dir doesn't exist in the topbase, then
it is attempted in the bottombases recursively. We
consider only pairwise layers; it is understood that
the stack is arbitrarily deep, and upon each iteration
through the stack the previous bottombase becomes the
new topbase.

Whiteouts:
if a file exists in both layers:
    if it is removed: remove from top, create a whiteout to hide bottom
if a file exists on top and not on bottom layer:
    if it is removed: remove from top
if a file exists in bottom and not on top layer:
    if it is removed: remove from bottom
if a file is whiteout on top and it exists in bottom:
if a file is whiteout on top and it exists in bottom:
    if it is removed: do nothing
if it is created: remove whiteout and create one on top
if it is to be accessed: FAIL

When an operation involves 2 file names:
- rename(from,to) gets called only if from and to are in this namespace.
- symlink(from,to) gets called only if from is in this namespace
  (to may or may not be in the name space)
- link(from,to) gets called only if from and to are in this namespace
*/

```

FIGURE 20

```

/* GROUP 1:
Operations on file that must exist.
getattr
readlink
chmod
chown
truncate
utime
read

Semantics:
fcn (path, args)
{
    GetTopPathState(path, NULL, &topExists, &isWhiteOut, &topPath);
    if (topExists)
    {
        // exists in top layer, use it
        return lowerFcn(topPath, args);
    }
    else
    {
        if (isWhiteOut)
        {
            // it's white out on top, FAIL
            return - ENOENT;
        }
        else
        {
            GetBottomPathState(path, NULL, &bottomExists, NULL, &bottomPath);
            if (bottomExists)
            {
                // doesn't exist on top, exists on bottom, use it
                return lowerFcn(bottomPath, args);
            }
            else
            {
                // doesn't exist on top or bottom
                return - ENOENT;
            }
        }
    }
}
*/

```

FIGURE 21

```

/* GROUP 2: =====
Operations on file that must not exist. Operation create the file.
mknod
mkdir

Semantics:
fcn (path, args)
{
    GetTopPathState(path, &topMatchLen, &topExists, &isWhiteOut, &topPath);
    if (topExists)
    {
        // exists in top layer, FAIL
        return EEXIST;
    }
    else
    {
        if (isWhiteOut)
        {
            // it's white out on top, remove without and perform operation
            DelWhiteOut(topPath);
            return lowerFcn(topPath, args);
        }
        else
        {
            GetBottomPathState(path, &bottomMatchLen, &bottomExists, NULL, &bottomPath);
            if (bottomExists)
            {
                // exists on bottom, FAIL
                return EEXIST;
            }
            else
            {
                // doesn't exist on top or bottom, create file on layer with deeper match
                if (topMatchLen >= bottomMatchLen)
                {
                    return lowerFcn(topPath, args);
                }
                else
                {
                    return lowerFcn(bottomPath, args);
                }
            }
        }
    }
}
*/

```

FIGURE 22

```

/* GROUP 3:
Operations on file if it exists, file created if it doesn't.
open
write

Semantics:
fcn (path, args)
{
    GetTopPathState(path, &topMatchLen, &topExists, &isWhiteOut, &topPath);
    if (topExists)
    {
        // exists in top layer, use it
        return lowerFcn(topPath, args);
    }
    else
    {
        if (isWhiteOut)
        {
            // it's white out on top, remove without and perform operation
            DelWhiteOut(topPath);
            return lowerFcn(topPath, args);
        }
        else
        {
            GetBottomPathState(path, &bottomMatchLen, &bottomExists, NULL, &bottomPath);
            if (bottomExists)
            {
                // exists on bottom, use it
                return lowerFcn(bottomPath, args);
            }
            else
            {
                // doesn't exist on top or bottom, create file on layer with deeper match
                if (topMatchLen >= bottomMatchLen)
                {
                    return lowerFcn(topPath, args);
                }
                else
                {
                    return lowerFcn(bottomPath, args);
                }
            }
        }
    }
}
*/

```

FIGURE 23

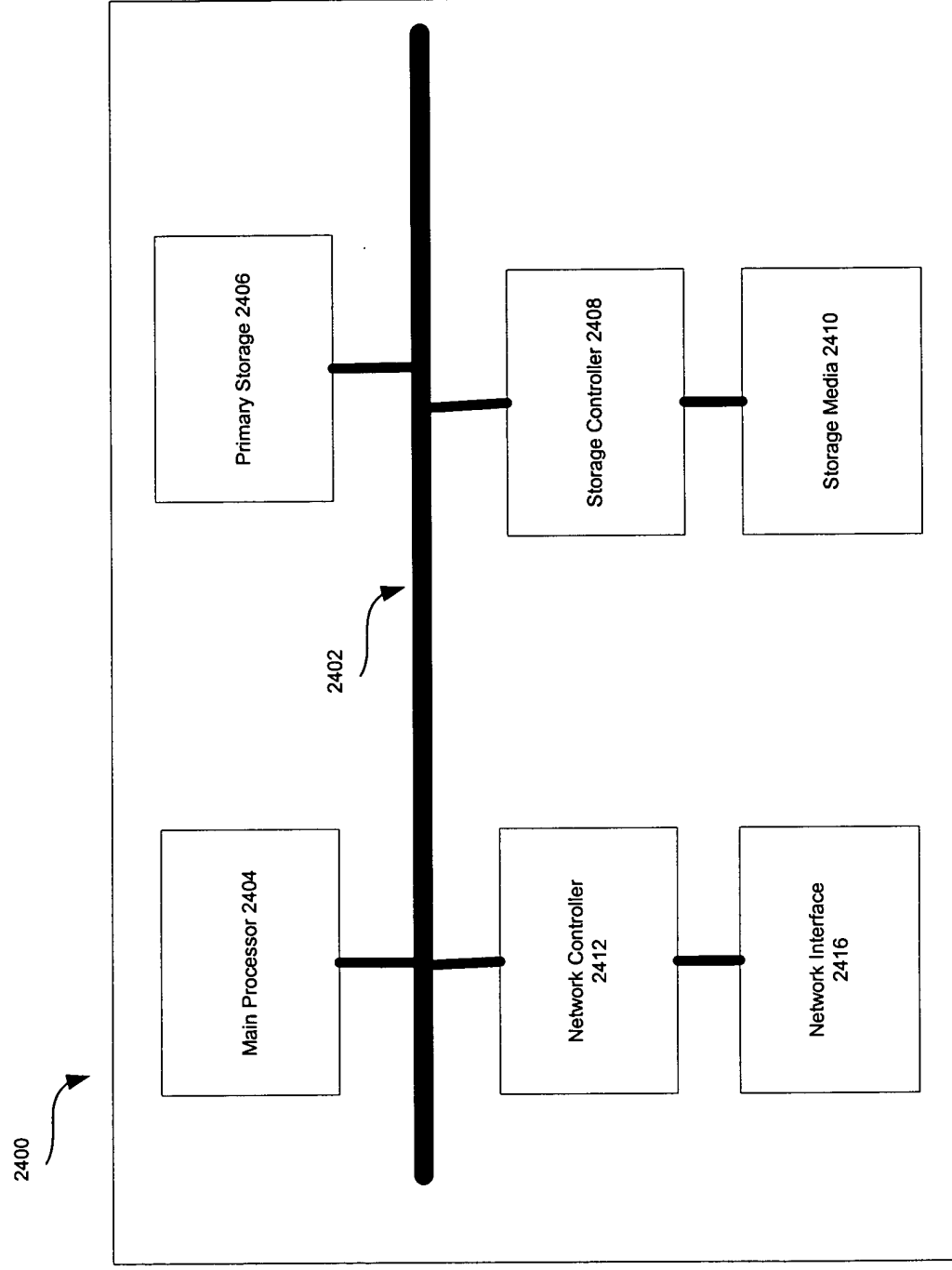


FIGURE 24